Cross-Layer Mischief in Networked Control Systems

Gregory Nazario, Michael Rosen, Lawrence Jackson, Michael Hankowsky
Carnegie Mellon University
Electrical & Computer Engineering
Pittsburgh, USA
{gnazario, mrrosen, lmjackso, mhankows}@andrew.cmu.edu

***Abstract*—SCADA (supervisory control and data acquisition) systems are becoming increasingly prevalent, especially wirelessly networked control systems. [citation] The integration of wireless networking into the operation of these SCADA systems is good for many reasons, but these integrations often ignore security as a priority. [citation] Our goal is to better understand the capabilities and the vulnerabilities these systems characteristically possess. In order to do this we will explore the use of cross-layer attacks on networked control systems and develop a framework for testing these attacks. We have developed testbed consists of a simulated network, a simulated factory driven by a matlab-ran Tennessee-Eastman process and the interface between these systems. Our research has confirmed that these vulnerabilities are exploitable and we have established this in our testbed. These vulnerabilities point to the necessity for a change in the way we look at securing SCADA integrated wireless networks.**

I. Introduction

The growing prevalence of SCADA systems have brought much needed attention to the class of cyberphysical threats. This paper examines the specific effects and testing of cyberphysical threats in wireless networking. Our intent is to simulate a networked control system similar to that of what one might find in a chemical plant. We approached Stuxnet destroyed a fifth of Iran's nuclear centrifuges. This is one simple example of SCADA SCADA Systems ... [more on scada]

A. The Framework

Subsection text here.

II. Related Work

There are a number of academic publications that we've referenced to gain a better understanding of the space we've developed in. The topic of cybersecurity has been examined in the following fields and we're going to go into more depth examining these works. We've looked at the general fields of: Networked Control Systems, Cyber-Physical Security, and Cross-Layer Mischief.

A. Networked Control Systems

The academic community has developed a number of research papers on the topic of security in networked control systems. However, there are two that are most relevant to learn from for our work. The survey of security in wireless sensor networks and the examination of adversary models for networked control systems. [12], [3]

B. Cyber-Physical Security

Ultimately, our work falls under the blanket of the concept of Cyber-Physical Security. This general topic has been explored more thoroughly by the academic community and there are many solutions that have been proposed to solved the problems the current model of security presents [11], [6], [2]. Specifically relevant to our concept of developing a MATLAB process driven network testbed is the work that has been done in process control security and the simulation of the Tennesee Eastman challenge.[9], [4] there has also been work done on developing various testbeds for the evaluation of cyber-physical systems.[1]

C. Cross-Layer Mischief

Our further focus on the affect of cross-layer attacks within networked control systems has yielded another large body of work to reference. Additionally, there has been work done to examine Networked control systems under these attacks. A number of attacks have been examined and surveyed on these systems, including DoS attacks in general[8] and more specifically in ad hoc networks [7], service authentication broken by timing attacks[5], and replay attacks [10]

III. Approach

Our goals were to create a modular framework that:

1) allows for extension by other developers, but without specific domain knowledge
2) provides for consistent parameterized simulation
3) allows for diverse simulations tailored to realistic SCADA conditions

A. Simulated Network

The network of our simulation is designed to give a real-time nature to the data transfer between the Tennessee Eastman(TE) physical simulation and the TE Controller. Both the controller and simulation are written in Matlab and connected to our network with the OMNeTBridge Class. (These are described in later sections.)

The network is based primarily on two factory level components, sensors and actuators, that send data to a remote facility that sends back control signals. This is based on most SCADA systems that need remote facilities to monitor their data. These signals are sent over a simulated ”Internet” that allows accurate packet loss, bit errors and data rates.

Due to limitations of the default Inet packets we are not actually sending the data through the network. Rather, the

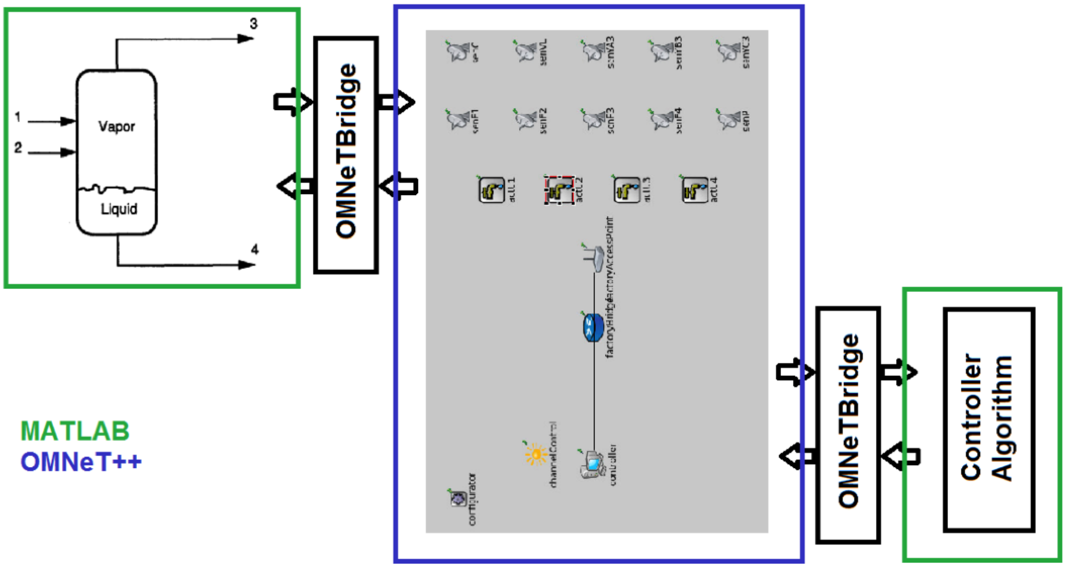

Fig. 1. System Diagram.

packets the controller receives act as a trigger to have the controller update the data from that sensor, and send an update to the actuators if need be.

The actual implementation of these components is described in later sections.

B. Matlab Simulation

We chose to use MATLAB simulation for the physical system and it's controls, as there are many chemical process models currently implemented in MATLAB. This allows for the ease of swapping out current MATLAB models for new ones.

1) Tennessee Eastman: In order to keep the complexity down, we decided to model the chemical process in the plant as a simplified Tennessee Eastman problem. Due to this approach, we looked to previous work in the field, and found many models already modeled in a hybrid of FORTRAN and MATLAB. However, this introduced more complexity, due to using three different programming languages to implement what should be simpler. Therefore, we would build a custom MATLAB class that contained the same functionality. This includes a both the simulation of the actual Tennessee Eastman system, and the steady state controller calculations.

IV. Implementation

A. Simulated Network

The network of our simulation is designed to give a real-time nature to the data transfer between the Tennessee Eastman(TE) physical simulation and the TE Controller. Both the controller and simulation are written in Matlab and connected to our network with the OMNeTBridge Class.

B. Controller

The controller is essentially a server module that takes in signals from each sensor, pulls the relevant data from the MATLAB simulation and sends it to the MATLAB based controller. This process allows us to accurately simulate the network passing packets while avoiding the headache that is implementing actual data passing packets in OMNeT. Th

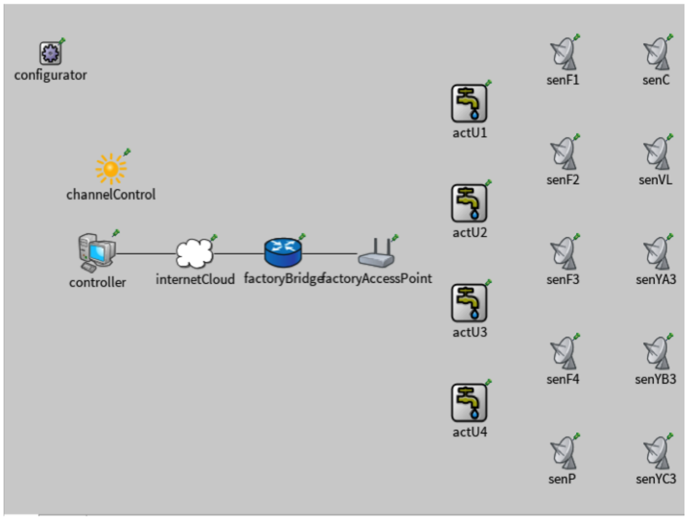

Fig. 2. Network Diagram

1) Actuator: The actuator acts as a server which receives an update packet from the controller, grabs its appropriate data from the controller OMNeTBridge and then passes it to the correct function. When these modules receive a data packet from the Controller IP address they then grab the current data from the Controller Bridge and pass it to the Simulation Bridge.

It is again a very simple module where most of the TCP application is focused on receiving a certain type of packet. These modules are also based on the INET "StandardHost" and are connected to the rest of the network wirelessly using 802.11 Wifi cards. These modules also lacked ideal implementation due to INET messages not being able to easily accept custom fields.

2) Sensor: The Sensor is a relatively simple module that sends an update signal based on a timer. This is to model the refresh rate of most sensors. We have assumed that all sensors and network controllable and are able to tirelessly connect to a local access point via 802.11 Wireless LAN.

These modules send a TCP packet to a known IP address which in turn triggers the controller to update. This is a non-ideal method of updating as it does not allow correct implementation of attacking the data on the network. Ideally INET's TCP packets would allow us to send data to them, however as some modules are currently implemented deep copies are not done of messages or inappropriate casts are used. Future work should be planned to modify these modules or find a more correct implementation of INET messages that allow custom data fields to be created and sent.

C. MATLAB Simulation

Our MATLAB both simulated a system and a controler of the Tennessee Eastman problem, and used a custom packet interface between C++ and MATLAB as explained below.

1) Tennessee Eastman: We built a custom simplified Tennessee Eastman model in MATLAB. This was based off of a preexisting FORTRAN model, but was re-implemented in MATLAB for ease of use and reduced complexity . Our model contains variable vectors, an input vector, a state vector, and an output vector. Then by using the equations provided in Ricker, the simulation simply takes in the input vector from OMNeT++, and outputs the output vector, which includes the sensor data, to OMNeT++ via the MATLAB bridge.

On the other side of the network, we also implemented a controller in MATLAB for the Tennessee Eastman system. This used steady state calculations to determine the optimal controls. The controller would input the output vector from the Tennessee Eastman system via the MATLAB bridge, and output a set of input vectors to be transmitted across the network into the MATLAB bridge.

D. C++ to MATLAB Bridge

In order to connect the two models implemented in MATLAB and OMNeT++, a bridge was implemented using standard TCP sockets. Both sides implement classes which makes communicating with the simulation running on the other program easy to interface with. Packets in the form of strings are sent back and forth. These packets contain a header and a variable number of data values. Each data value has a name as a string, a type, and a value (the actual representation of which depends on the type). The supported types are integers, floats and strings, with doubles being partially implemented. However, as the Tennessee-Eastman simulation only used integers and floats, only these types have been significantly tested. Both sides parse the packets into their respective language representations to allow for easier use.

On the MATLAB side, a MATLAB class, OMNeTPipe, is used to establish a connection and send and receive packets. In the constructor for this class, a TCP server socket is opened on a specified port. Once connected to OMNeT++, the pipe can be used to send and receive packets, where the receive function is blocks until a full packet is received. Once a packet string is received, the string is parsed and the header is returned as a string with the data values stored as a map, with the variable name used as the key. When sending a packet, the send function takes in a variable number of arguments, where the header and data names, types and values are passed into the function. These are then formed into a string and written to the socket. The send function is non-blocking.

On the OMNeT++ side, a few C++ classes are used to complete the interface. The first, OMNeTPipe, mirrors the MATLAB side, which, in the constructor, attempts to connect a TCP socket to a given port and host. A socket exception is thrown on failure. The socket interface for the C++ side was borrowed from Jeff Donahoo's PracticalSocket class from Baylor University School of Engineering and Computer Science. Once a connection is established, similar send and receive functions can be used to transfer values to and from MATLAB. However, the C++ interface also uses another class to make the storage of packets more convenient. The OMNeTPk class is used to store values to be sent and received from MATLAB. Thus, the send function takes in a pointer to one of these objects, and the received function returns a pointer to an OMNeTPk object. These objects allow values to be set with the addVal function and values to be retrieved by name. Internally, the OMNeTPk stores values, names and types in C++ vectors, using void* generic values to store the integers, floats and strings supported by the pipe interface. Thus, similar to the MATLAB map, the variables can be retrieved via their names associatively.

In order to better facilitate communication between OMNeT++ and MATLAB in the Tennessee-Eastman simulation, another C++ class was implemented. The OMNeTBridge class is a more project-specific class which holds the various control and feedback variables from the MATLAB simulation for the various OMNeT++ network objects to use. This class uses the OMNeTPipe to connect to MATLAB to send and received data and store it locally. Thus, a system state is maintained in both MATLAB and OMNeT++. On the system side, a bridge connects the MATLAB Tennessee-Eastman simulation to the sensors and actuators. On the controller side, a separate bridge connects the MATLAB controller model to the controller network application. In the constructor of the bridge, an OMNeTPipe is opened depending on the type of the bridge (controller or system side). Once established, an UPDATE packet is sent. This packet consists of an id (always 1) and a dt value. OMNeT++ is used to keep time, thus all packets sent to MATLAB contain this change in time so the MATLAB model can update itself appropriately. MATLAB, after updating the model, sends back a STATUS packet containing s the current settings for all 14 variables. The bridge then stores these values. The bridge also provides methods to set and request parameters from the MATLAB model. When requesting data, the bridge sends MATLAB an UPDATE packet with the time change since the last packet was sent over the bridge and then waits for the STATUS response. Once received, the bridge updates its internal state and returns the requested value to caller (being a sensor or the controller). In the case of a setVal call, the bridge sends a CHANGE packet, which includes the information from an UPDATE packet with whatever parameters are to be set to the given values. Once sent, a STATUS packet is received to update the internal state of the bridge. Depending on the bridge type, either control parameters (for actuators) or status parameters (for the controller) are sent. This bridge object makes interfacing MATLAB and OMNeT++ simulations quick and straightforward.

V. Challenges

A. Omnet++

B. Matlab

C. Bridge

VI. Evaluation

A. Metrics

We will consider three metrics when evaluating the results of our experimentation: sensor data, control signals and packet throughput.

1) Sensor Data: We consider Sensor Data because this includes crucial data about what is actually occurring in the simulated plant process (Tennessee-Eastman, in this case).

2) Control Signals: We consider the Control signals because these signals are meant to be directing the process, and how they react when parameters of the simulation are changed a crucial to our understanding of how various attacks effect our simulation.

3) Throughput: We use the packet throughput as an evaluation metric because this data is extremely relevant in denial of service type attack we have implemented.

VII. Results

A. Baseline

B. DDoS

C. Spoofing

VIII. Discussion

The project was relatively successful with the completion of the simulation framework and the implementation of some basic attacks. While we would have liked to include more attacks and improved the framework to more accurately model modern SCADA networks, the construction of the communication bridges and basic example of a network was a significant project and can be expanded to larger models and simulations if needed. The pipe and bridge system can used used for arbitrary MATLAB or OMNeT++ models, thus our example of a distributed SCADA network controlling a Tennessee-Eastman process is just one example. The framework was constructed to be dynamic, thus allowing for plug and play or various network and MATLAB models.

In the course of development, several challenges impeded work on the framework. Working through C++'s abandonment of the standard cast and void* generic type from C was somewhat frustrating, meaning converting floats became difficult, as well as dealing with development for both 32 and 64-bit machines. Aside from these minor technical issues, some of the biggest problems arose from the way in which INET and OMNeT++ model data transfer. As mentioned about, OMNeT++ and INET only attempt to create a timing-accurate model for various network protocols and hardware. Data-transfer is relatively absent and thus, when sending packets across the network model containing data from either the sensors or controller, the actual values being sent where often lost. Thus, ensuring data was able to get through the network was a significant challenge as the INET framework is very large and finding the few lines which destroy and recreate the packet was difficult. While we did get the simulation running, INET should be capable of preserving objects across the network, and it is a mystery why it does not.

While we did not model an exact SCADA system with industry software, the dangers illustrated in our results are still of significant concern. Throwing the process into an unstable state was not difficult and control system in general require real-time control. Delaying or spoofing signals in such systems can lead to instability. In the case of nuclear power or other chemical processes, these instabilities can be disastrous. As more and more control systems find their way onto the Internet, remote attacks become easier and physical access to the plant or system becomes unnecessary. Attackers from across the country or even other nations can target and disrupt control systems for critical infrastructure. Thus, a significant threat exists and more work needs to be done to ensure the safety and security of the various critical systems now on the Internet.

IX. Future Work

A large goal of this project was to develop a framework that would enable further research to be done in this space. What we've provided makes it very easy to develop different network configurations for corresponding process simulations in Matlab. There are three major fields of further work that we propose: extending the framework, experimenting with more attacks and validating configurations.

A. Extending the Framework

The framework is functionally complete, but there are additions that could be made. More modules could be adapted to support a greater range of network simulations. The addition of primitives to support cryptography, or at the least simulated cryptography, would enhance the realism of this framework and allow for truer to life attacks.

B. Experimentation

A large body of future work could easily consist of just experimentation with different types of misbehavior and exploitation.

C. Validation and Research

It would be valuable to do further research into the common configurations of networked control systems in industry. This information would certainly help focus future experimentation and development.

X. Conclusion

We have developed a

ACKNOWLEDGMENT

The authors would like to thank Bruce DeBruhl for guiding our research and for helping us to refine our ideas.

REFERENCES

[1] Siaterlis et al., *EPIC: A Testbed for Scientifically Rigorous Cyber-Physical Security Experimentation*, Vol 1, No.2 IEEE Transactions on Emerging Topics in Computing, 2013.

[2] Mo et al., *Cyber-Physical Security of a Smart Grid Infrastructure*, Vol. 100, No.1 Proceedings of the IEEE, 2012.

[3] Cardenas et al., *Challenges for Securing Cyber Physical Systems*, 2009.

[4] N. L. Ricker, *Model predictive control of a continuous, nonlinear, two-phase reactor*, Seattle, WA: University of Washington, 1993.

[5] A. Hayes, *Network Service Authentication Timing Attacks*, IEEE Computer and Reliability Societies, 2013.

[6] H. Beitollahi, G. Deconinck, *A Dependable architecture to mitigate distributed denial of service attacks on network-based control systems*, Vol.4, International Journal of Critical Infrastructure Protection, 2011.

[7] S. Radosavac, N Benahammar and JS Baras, *Cross-layer attacks in wireless ad hoc networks*,Princeton, New Jersey: Conference on Information Sciences and Systems, 2004.

[8] S. Amin, A. Cardenas and S.S. Sastry, *Safe and Secure Networked Control Systems under Denial-of-Service Attacks*, Berlin, Germany: Springer-Verlag, 2009.

[9] Hashimoto et al., *Safety securing approach against cyber-attacks for process control system*, Vol. 57,Nagoya, Japan: Computers and Chemical Engineering, 2013.

[10] Y. Mo and B. Sinopoli, *Secure Control Against Replay Attacks*, 47th Illinois, USA: Annual Alerton Conference, 2009.

[11] S. McLaughlin, *Securing Control Systems from the Inside: A Case for Mediating Physical Behaviors*,IEEE Security & Privacy, 2013.

[12] X. Chen, K. Makki, K. Yen and N. Pissinou, *Sensor Network Security: A Survey*, Vol. 11, No. 2IEEE Communications Surveys & Tutorials, 2009.